

Coinmonks

Bitcoin + Forensics + Cryptocurrency + Blockchain + Crypto +

Free Bitcoin Forensics - Part 2



Pagan

Follow

23 min read · Dec 13, 2020



21



Welcome to part 2 of "Free Bitcoin forensics", you can find the first part of the article [here](#). In this part, I'll be exploring some more advanced techniques and concepts of BTC forensics. The following topics will be explored in this article:

- Transaction graph
- Blockchain techniques for evasion/detection
- Wallet Clustering
- Wallet Fingerprinting
- Scripts and address formats
- CoinJoins and PayJoins
- Other misc identification patterns
- The future of privacy

So without further ado let's dive in:

Transaction graph

To be effective at Bitcoin forensics one must learn to use a visualization tool when tracking transactions, that is the only that way the big picture view of transactions can be seen. The only free tool I am aware of that can be used for visualization is oxt.me which allows seeing transaction path like this:



<https://oxt.me/graph/transaction/tiid/3124363752>

Open in app ↗

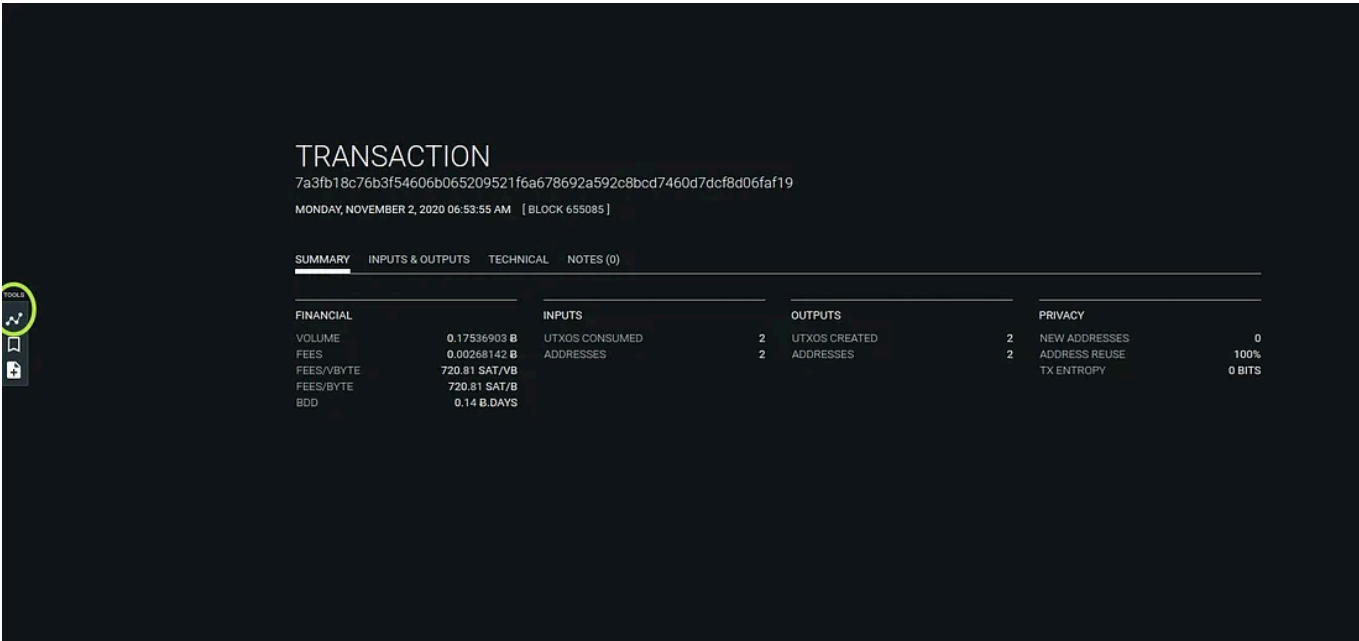
Sign up

Sign in

Medium

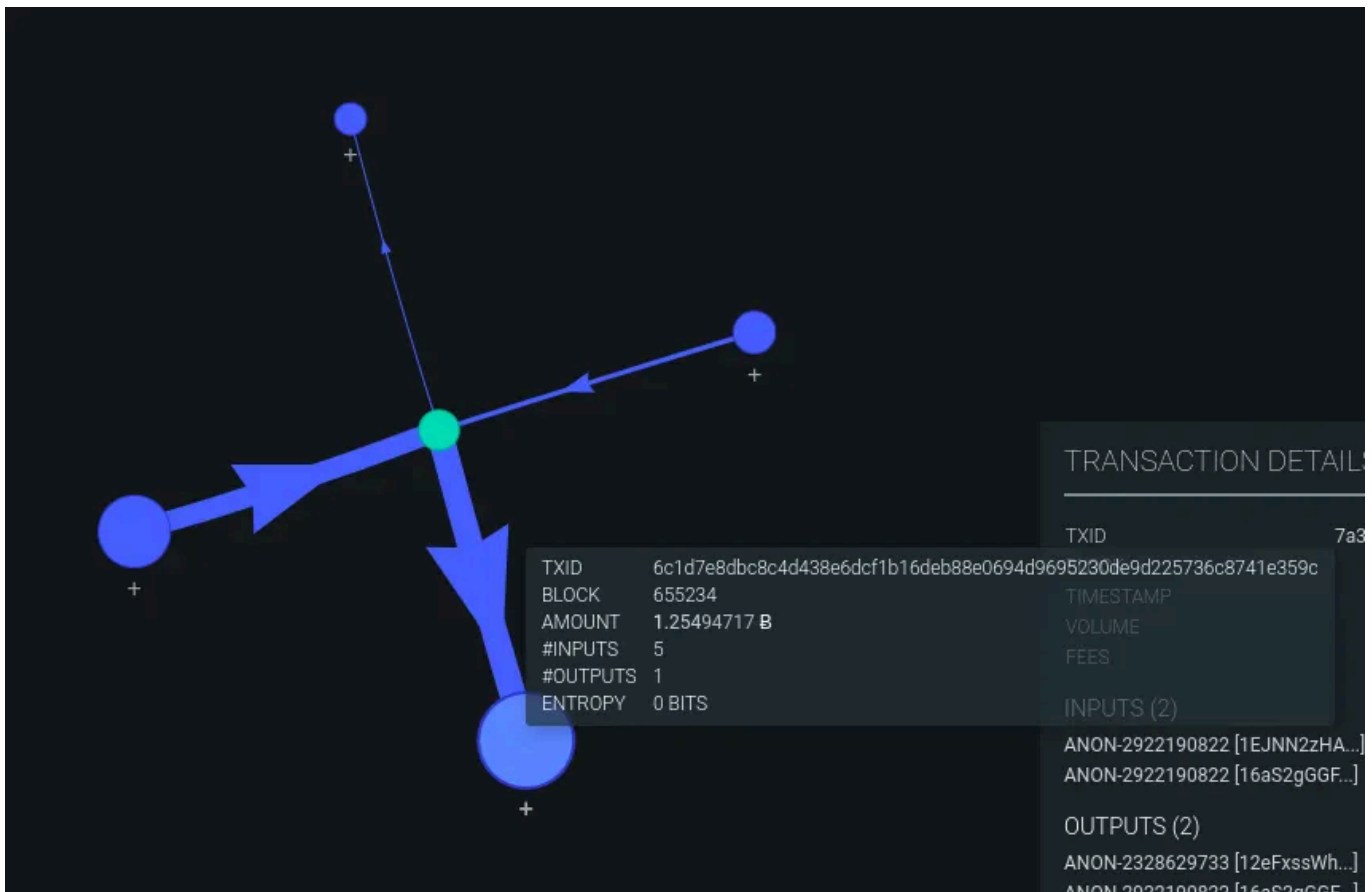
Search

Write



<https://oxt.me/transaction/7a3fb18c76b3f54606b065209521f6a678692a592c8bcd7460d7dcf8d06faf19>

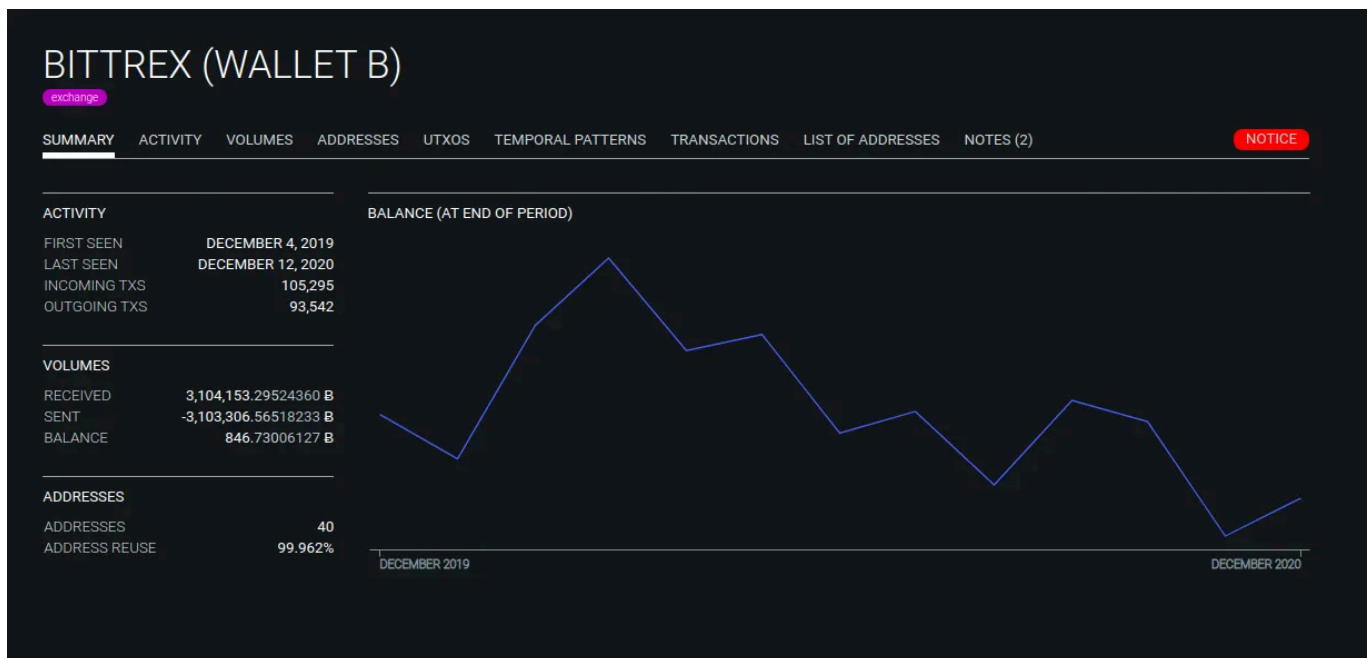
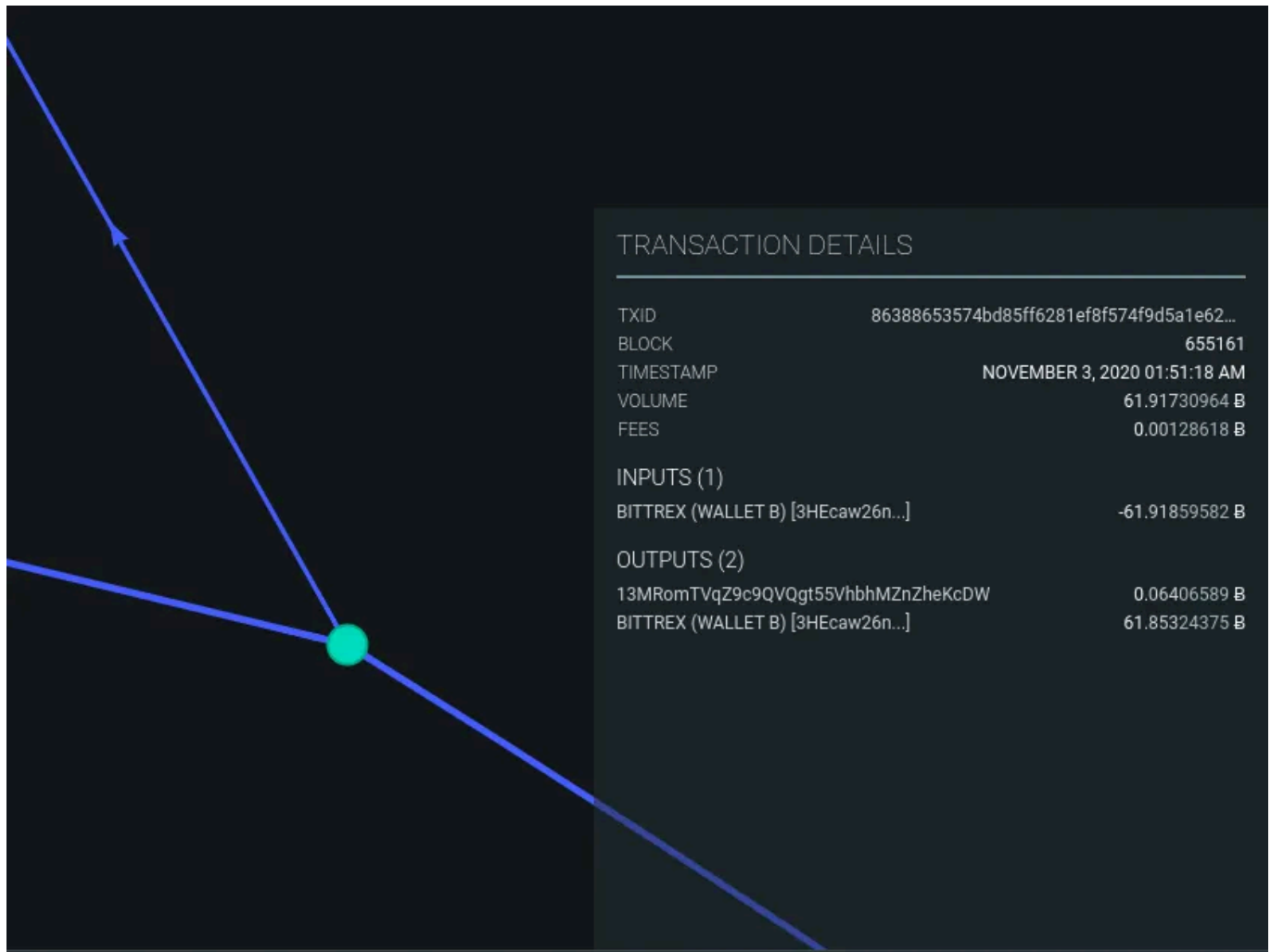
Then expand the transaction graph as you see fit:



The transaction graph can grow as large as you wish:



[oxt.me](#) supports wallet clustering (attempting to determine if multiple addresses belong to the same entity) as well, the website does it automatically, so if you see a wallet name next to an address (they may be called ANON... or have a name of an exchange if it has been indexed) simply click on it to see wallet information:



<https://oxt.me/entity/tiid/2556015810>

I recommend creating an account with oxt.me to gain access to extra features such as saving transaction graphs, leaving comments, and being able to see all the addresses associated with specific wallets.

Visualizing transaction graphs like this can give you an amazing big picture view and allow you to inspect many transactions much quicker (especially if those transactions are repetitive), but you should be mindful of small picture individual transaction view as well and make sure not to miss coinjoins or exchange transactions, correctly recognize change addresses and so on, otherwise you'll end up just mindlessly clicking bubbles instead of learning something from the visualization.

research.oxt.me beautifully utilizes the visualization to aid them in their super in-depth investigations, they are a great example of how this sort of tools is used in practice: [publications](#)

Some interesting transaction patterns

Now, that we're familiar with the transaction graph let's take a look at a few common/interesting transaction patterns that are used to increase or destroy privacy and how they would look on the graph.

Peel chain — a series of transactions usually with one input and 2 outputs where one of the outputs (usually the smaller one) is "peeled off" to be deposited to a third party (such as an exchange). The other output is the change that continues the "peel chain" in subsequent transactions.

The peel chain method has been known to be used by hackers who wished to make their funds harder to track and hide the origins of the funds from exchanges in which they want to launder the funds. The concept of "Peel chain" was even mentioned in an official United States complaint against

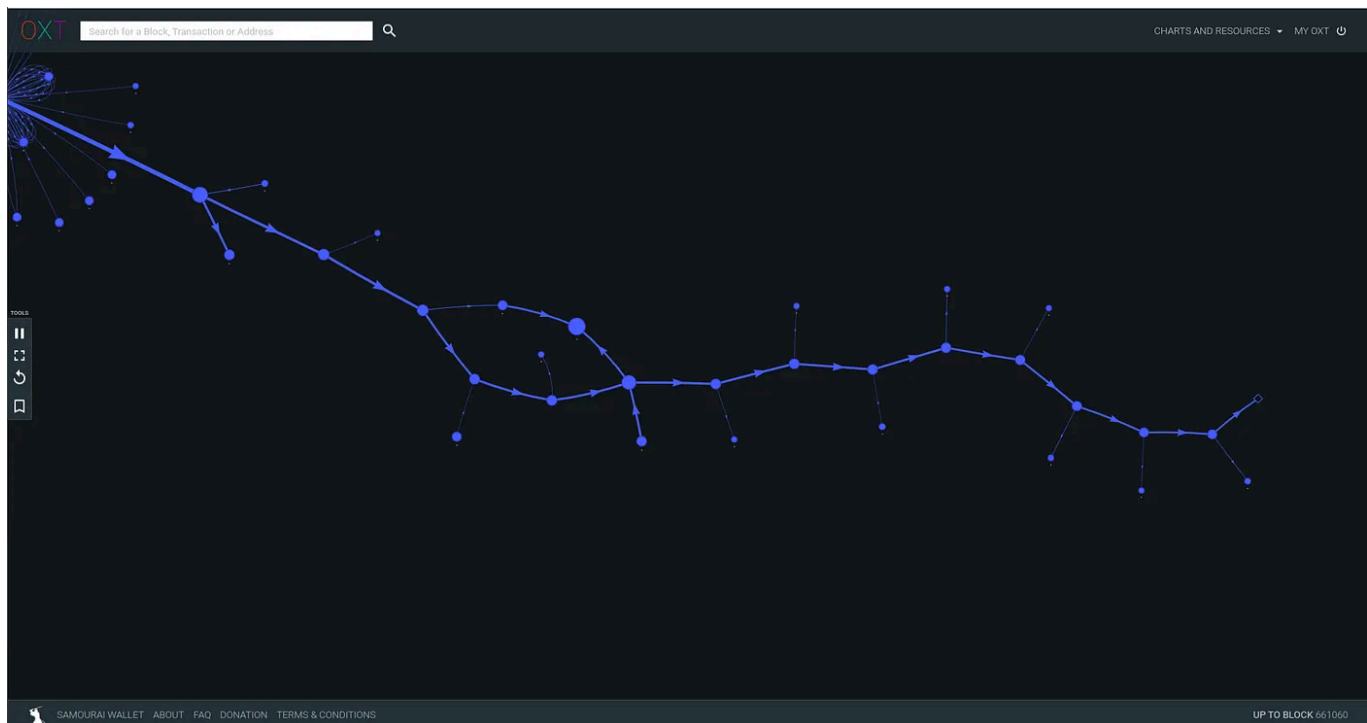
individuals suspected of being involved with several exchange hacks:

<https://www.justice.gov/opa/press-release/file/1253491/download>

The use of peel chain was also utilized by Twitter hackers when they tried to obfuscate the movement of their stolen funds:

<https://ciphertrace.com/twitter-hack-update-blockchain-analysis-helps-identify-twitter-hackers/>

A peel chain when visualized via oxt.me will look something like this:



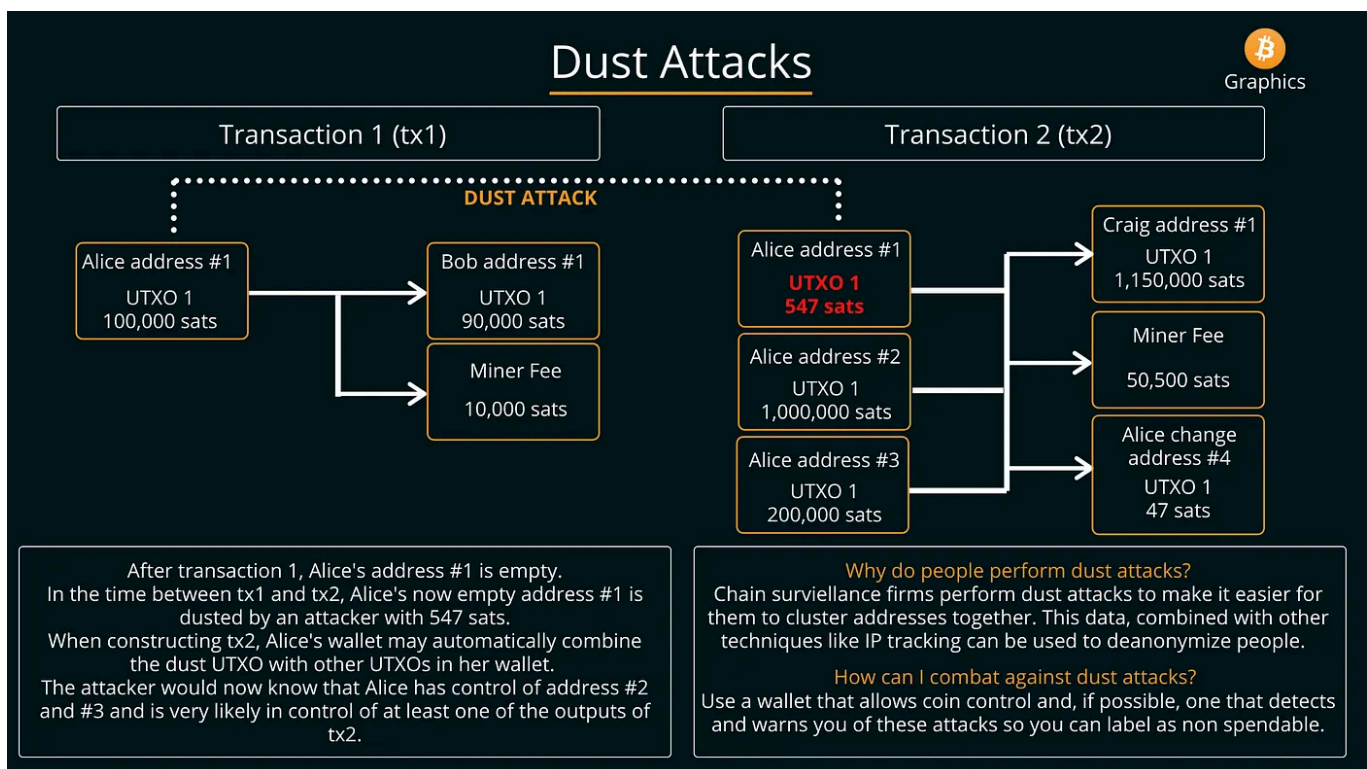
<https://oxt.me/graph/transaction/tiid/2923573673>

Generally, I would say that peel chains are rather ineffective in hiding stolen funds or making the stolen funds harder to track, though it may be a bit time consuming, a good investigator will be able to track peel chain transactions.

Dust attack — the name “dust attack” refers to a special kind of attack where the attacker sends a small amount of BTC to (usually empty) addresses of the

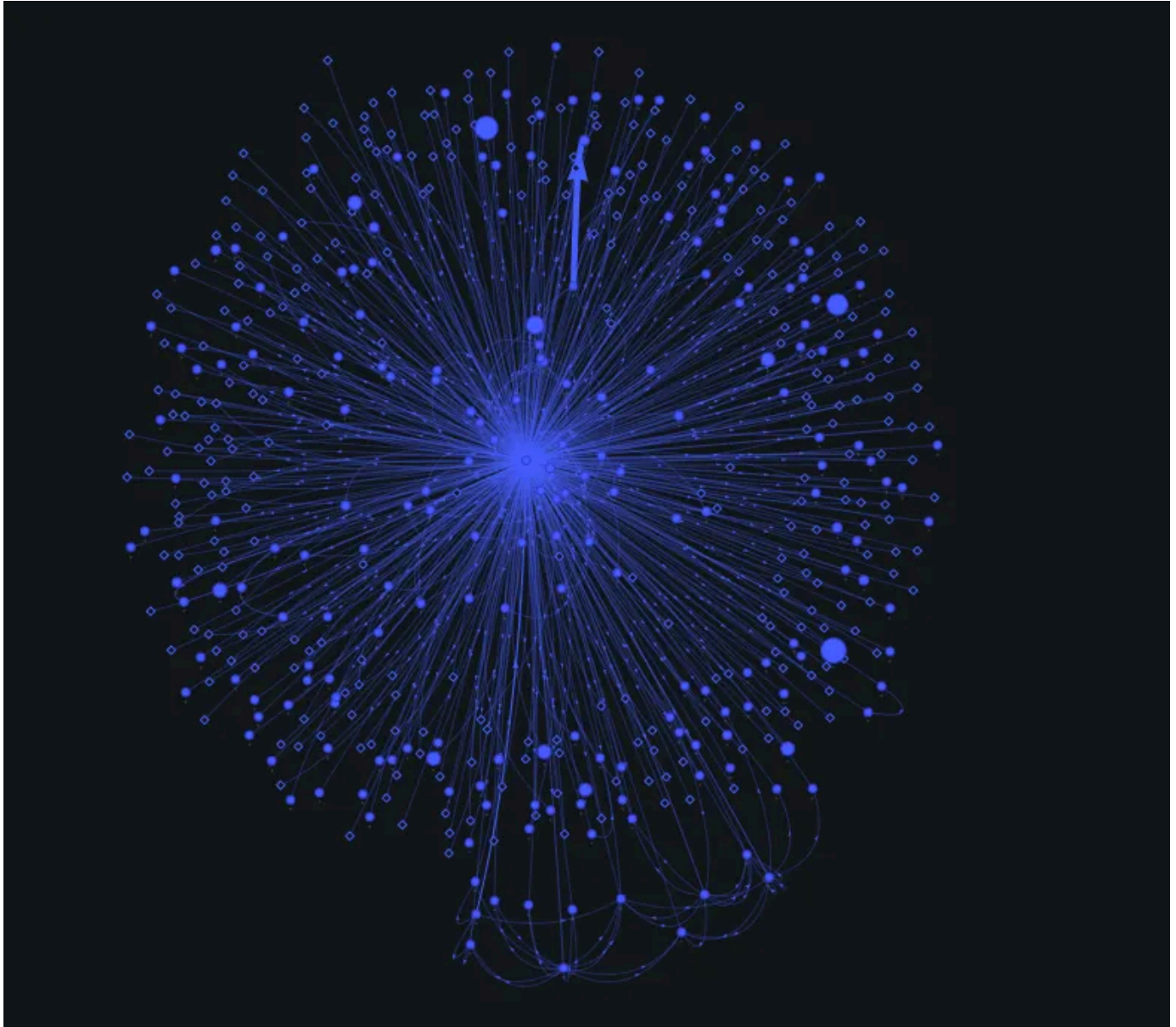
entity or entities they wish to track in the hopes that the entity will fail to notice the small new deposit to their addresses and will inadvertently end up using the funds in that address in another transaction, therefore, revealing more of their previously unknown addresses. This attack works because most Bitcoin wallets do not offer the option for the user to manually select the unspent outputs they wish to spend but does that automatically and makes no distinction between the "dust" unspent outputs and the regular ones.

If this explanation does not help, please see the following explanation graph from [bitcoingraphics.page](#) I find the graph very helpful for understanding the attack:



Dust attacks are widely known in the Bitcoin sphere due to the fact that there have been many large scale dust attack through-out BTC history. Here are a few examples of dust attack transactions:

- [c61e8ee41d8317dc0059a4fa2cbbd5b55c75c9aac521c8b3d47b763f2d897b37](https://oxt.me/transaction/c61e8ee41d8317dc0059a4fa2cbbd5b55c75c9aac521c8b3d47b763f2d897b37)
- [c9f5ee83627560c02c5da5f52190f83d3c5ef46514be3bcbfc83365cf6fea2d7](https://oxt.me/transaction/c9f5ee83627560c02c5da5f52190f83d3c5ef46514be3bcbfc83365cf6fea2d7)
- [c61e8ee41d8317dc0059a4fa2cbbd5b55c75c9aac521c8b3d47b763f2d897b37](https://oxt.me/transaction/c61e8ee41d8317dc0059a4fa2cbbd5b55c75c9aac521c8b3d47b763f2d897b37)



<https://oxt.me/transaction/c61e8ee41d8317dc0059a4fa2cbbd5b55c75c9aac521c8b3d47b763f2d897b37>

The oxt.me visualization of dust attack transactions is quite a thing to behold...

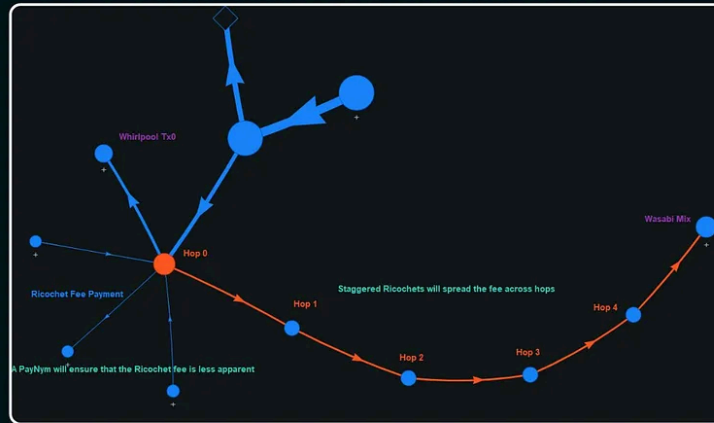
Transaction hopping — Besides the peel chain, another method to hide the origin of funds from exchanges is by simply sending them through a series of transactions to newly-created addresses (that you, of course, own). It is common knowledge that exchanges hire blockchain forensic firms to determine the source of the funds that arrive at their exchange, but the software that those forensics firms use likely only scans the last 3–4 transactions the upcoming funds have been involved in (please note that I'm just speculating here I don't know for sure how many of the last transactions are scanned and surely that differs between different blockchain forensic firms).

You can of course do this sort of transaction hopping manually simply by sending your funds to your newly created unused addresses a few times, but there are also some wallets like Samurai wallet that support this feature natively: [ricochet](#)

A wonderful explanation of how the transaction hopping function (called "Ricochet" in the wallet) is again presented at <https://bitcoingraphics.page/>:

Ricochet Transaction

Example from oxt.me



Attributes

- Creates additional 'hops' between transaction origin and destination
 - Creates deniability
 - Can be regular or 'staggered'
- Staggered guarantees each hop is in a different block

When to use it?

- For any type of spend where increased 'hops' of history are required
- Useful for sending to exchanges or to another regulated entity

Limitations

- 200,000 sat fee
- Connect to Samourai PayNym for greater privacy before paying the fee

Wallet Clustering

By this point, you may have asked yourself how do block explorers such as oxt.me and walletexplorer.com cluster wallets? Although I cannot be 100% sure how individual explorers guestimate wallets, there is certainly no magic there, they are certainly using the simple assumption that all inputs in a transaction belong to the same wallet and likely using a few tricks to determine the likely change addresses. Anyone can do the same manually for the entire blockchain using any transaction as a starting point. Here is how I would do that:

Senders		Recipients	
← ⓘ 0.00087861 BTC	39mrBKf2foBLcoy92jyct7HAbfdg3y6dyX	3NMNZK6Z27aERLkamPrz5HNce8pMSiVbr9	0.00256045 BTC Unspent
← ⓘ 0.00312243 BTC	3JcSGFetd5R5QM8dgG6BvhmF5v2S4tTR4L	37HFBmZUmMPEN8ihv28fTwMyXVPQv16kp7	0.00098375 BTC ⓘ →

<https://blockchair.com/bitcoin/transaction/d681640cfcbcec4b0b6ede9ce9aaaa6106158956e89cf3b5ed4f7e09b40b2630>

Let's say we begin with the transaction above. First of all, we can immediately assume that both the input addresses in this transaction belong to the same wallet due to the common input ownership heuristic. With this knowledge the next logical step is to explore them a bit deeper:

- [39mrBKf2foBLcoy92jyct7HAbfdg3y6dyX](#)
- [3JcSGFetd5R5QM8dgG6BvhmF5v2S4tTR4L](#)

In the history of address [39mrBKf2foBLcoy92jyct7HAbfdg3y6dyX](#) we can see that this address was involved in another transaction where it was among the deposit addresses:

Senders		Recipients	
← ⓘ 0.00098375 BTC	37HFBmZUmMPEN8ihv28fTwMyXVPQv16kp7	3Ev4AbVvW9BQBsYJVCERBHMurTaEvJq13a	0.00750000 BTC Unspent
← ⓘ 0.00001275 BTC	38GBwJUXVcBVwysEmJZLLGPCfY8DF4wSAg		
← ⓘ 0.00102703 BTC	3Jo5vrdzCQtP998EkuUCvczTeAAe3nc5VT		
← ⓘ 0.00081422 BTC	39mrBKf2foBLcoy92jyct7HAbfdg3y6dyX		
← ⓘ 0.00106890 BTC	31ivvBNuFuMF7Uf17HWePb4XUtFNDNAiSJ		
← ⓘ 0.00220400 BTC	38CRckJfg4t42ziAatuZw7MwRWVhY65Rg		
← ⓘ 0.00100000 BTC	3NuwtQodHj4F893DtMokkB5vxUR4rGYrcu		
← ⓘ 0.00051954 BTC	3QnL2m2osNDUudT8dkcqWjzSAE4Bh1WqEb		
← ⓘ 0.00124037 BTC	39mrBKf2foBLcoy92jyct7HAbfdg3y6dyX		

<https://blockchair.com/bitcoin/transaction/8d6097bc1f743b8c280da44ca8d604c4fccd2c65532f521262fd6eff20ddce79>

Since this transaction does not look like a Coinjoin we can make the assumption that all the input addresses here also belong to the same wallet. So now our wallet has grown to the following addresses:

- [39mrBKf2foBLcoy92jyct7HAbfdg3y6dyX](#)
- [3JcSGFetd5R5QM8dgG6BvhmF5v2S4tTR4L](#)
- [37HFBmZUmMPEN8ihv28fTwMyXVPQv16kp7](#)
- [38GBwJUXVcBVwysEmJZLLGPCfY8DF4wSAg](#)

- 3Jo5vrdzCQtP998EkuUCvczTeAAe3nc5VT
- 31ivvBNUFuMF7Uf17HWePb4XUtFNDNAiSJ
- 3NuwtQodHj4F893DtMokkB5vxUR4rGYrcu
- 3QnL2m2osNDUudT8dkcqWjzSAE4Bh1WqEb
- 39mrBKf2foBLcoy92jyct7HAbfdg3y6dyX

Next, I would check if the new addresses the same way to see if they are involved in other transactions as inputs and if they add any newly discovered addresses to the list, check them, add newly found input addresses to the list, and so on until all such addresses have been found and added to the wallet address list.

With this technique alone good approximations of wallets can be made, but the approximations can be further improved if change addresses can be detected and added to the wallet address list as well. The newly detected change addresses could then also be checked to see if they are involved in any other transactions as inputs and any addresses that are also inputs in that transaction added to the list.

Finally, being able to detect coinjoins would be helpful with wallet guestimation, since, obviously coinjoins do not follow the common-input-ownership heuristic.

All of this, of course, is very automatable and it's fair to assume that both oxt.me and walletexplorer.com use some combination of mentioned techniques to determine wallets.

Wallet fingerprinting

Each BTC transaction is unique and most of them differ from one another quite significantly when looked at closely. In fact, they differ so much that in certain situations it can be possible to determine from what wallet the transaction was made. To illustrate this I have inspected several wallets and made this table with some of their characteristics:

Wallet	Supported Address Type	Allows to set a Fee?	Change Address Format	Reuses Address for Change?	RBF?	Accepts Segwit?
Atomic	Legacy	Yes	Legacy	Yes	No	Yes
Jaxx	Legacy	No	Legacy	No	No	No
Guarda	Legacy	Yes	Legacy	Yes	No	Yes
Mycelium	Segwit/Legacy/P2SH	Yes	Segwit	No	No	Yes
Edge	Legacy/P2SH	No	P2SH	No	No	Yes
Green wallet	P2SH	Yes	P2SH	No	Yes	Yes
Enjin	Legacy	Yes	Legacy	No	No	No
Coinomi	Legacy/P2SH	Yes	Legacy	No	No	Yes

Given a few transactions belonging to one of these wallets it would not be hard to determine to which wallet those transactions belong to (unless the transactions belong to Atomic or Guarda wallets, since those have similar characteristics).

Though from these examples wallet fingerprinting can look trivial, in reality, however, it is substantially complicated by the fact that there are hundreds of cryptocurrency wallets (although let's be real here, only some of them are popular enough to pay attention to) and those wallets change constantly and with the changes in the wallets, their transactions also change. Any entity who wishes to fingerprint wallets effectively would not only have to keep track of a large number of wallets but also keep up to date with the changes in those wallets. To my knowledge no free tools attempt anything like this, I

do not know if professional blockchain forensic tools (that are used by companies like Chainalysis) are capable of this, but I assume that if not, they will start fingerprinting at least some wallets at some point in the future.

The characteristics mentioned in the table are not the only ones that can be used to identify a wallet. If you are interested in a more expanded list of characteristics that could be used to identify a wallet I'd recommend reading the following section of the Bitcoin privacy wiki:

https://en.bitcoin.it/wiki/Privacy#Wallet_fingerprinting

Scripts and address formats

In the previous section, we showed how transactions can have some easily noticeable differences that help to determine what wallets they may belong to. This is great and all, but easily noticeable will only get you so far, to understand the full uniqueness that Bitcoin transactions can have as well as common patterns between them, we need to take a closer look at transactions. This section will attempt to do exactly that, but before we begin I would highly recommend reading the following chapters of Mastering Bitcoin (if you haven't already), otherwise, it will be much harder to fully understand the information presented here:

<https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch06.asciidoc>

<https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch07.asciidoc>

To a casual Bitcoin user, it may seem that there are 3 types of Bitcoin addresses (the ones that start with a "1", the ones that start with a "3" and the ones that start with "bc1"). But in another, more correct way, there are 6 types of Bitcoin addresses. They become apparent once you start exploring "raw transactions", that is, transactions the way they really look before a

blockchain explorer makes them easier to follow. A good overview of what a raw transaction consists of can be found [here](#).

Before we dive into “raw transactions” themselves, however, let’s briefly discuss all address types:

P2PKH (Pay to Public Key Hash) — The original Bitcoin addresses that start off with “1”. They are quite simply the hash of the public key (hence the name). When Bitcoin was first started this was the only address type available and thus it is now the most widely adopted, with virtually, all exchanges, wallets, and other Bitcoin businesses accepting and supporting P2PKH. However, this address type is very simplistic and does not allow doing anything interesting with the address (such as making multisig transactions), thus, soon after Bitcoin started working a new type of address was introduced — P2SH.

P2SH (Pay to Script Hash) — Introduced in 2012, the P2SH format of Bitcoin address is an incredibly flexible one that allows embedding spending condition into an address (Bitcoin protocol supports a few hundred opcodes from which spending conditions can be created, in practice the most common novel use of P2SH is multisig), the spending condition for the output is hashed and thus baked into the address. Since the address only contains the hashed spending condition (and the hash can’t be reversed), the spending condition itself will only be revealed once the output is spent in what’s called a “redeem code”. If you are interested in learning more about how P2SH addresses can be created in practice I’d recommend reading [this](#) fantastic article.

P2WPKH (Pay-to-Wrapped-Script-Hash) and P2WSH (Pay-to-Wrapped-Script-Hash) — In 2017 the SegWit upgrade was introduced to the Bitcoin protocol

to improve the scalability and fix transaction malleability in Bitcoin. As part of this upgrade a new type of address, commonly referred as Segwit or Bech32 address, was introduced. Bech32 addresses can really be subdivided into 2 distinct addresses, the P2WPKH, and the P2WSH addresses, although they look similar at first glance (both start with "bc") they really are as different as P2PKH and P2SH. P2WPKH is the Segwit equivalent of P2PKH while P2WSH is the equivalent of P2SH. We will soon be taking an in-depth look at the differences between these addresses under the hood.

P2WPKH nested in P2SH and P2WSH nested in P2SH — together with Bech32 addresses, the Segwit upgrade also introduced a clever way to "hide" Bech32 addresses in a P2SH address. The advantage of doing so is compatibility, many exchanges and wallets still do not support sending BTC to Bech32 addresses, but virtually all support sending funds to P2SH addresses.

Now, let's take a look at "raw transactions" and how these address types look there. At it's most "raw" transactions will look something like this:

```
010000000144f9987226fd388e994446557e697507be0431cceb8ca50c8f26b59f575
7fe9f000000008b483045022100df173cf465c319888425b5210e306bdbf92e7a1bfe
861626fb98306d9d6e676902204d5e8019f4e2a783e99a2cbe46fe8b33679bc1af58
3ceaa99418585bc99c2ad4014104e6e67eeb906914a27d075003367f3e10d11b693
b9d30cdd912a70e94e3fdc59f50f0d61c8e03ebac6f440fdc4d1601fd699fb8f227cd
1cf15e1cd07acc83c3a1ffffffff0170830700000000001976a9143f81bb4512b6b403
caf3fe658c90a287359164b588ac00000000
```

But this is no good, we gotta decode this transaction from hexadecimal to a more readable format. The same "decoded" transaction can be viewed here: [720edc9c452709dbc1d49b93babaef8dd5fb813c3c9f63f880c1fff17061182f](#)

```

{
  "ver": 1,
  "inputs": [
    {
      "sequence": 4294967295,
      "witness": "",
      "prev_out": {
        "spent": true,
        "spending_outpoints": [
          {
            "tx_index": 0,
            "n": 0
          }
        ],
        "tx_index": 0,
        "type": 0,
        "addr": "1AEqTxEGtCLsdG8bagtxKDW6tnULyV276V",
        "value": 600000,
        "n": 0,
        "script": "76a9146556fee150168a4708c2e4db5a5232b6a0910a88ac"
      },
      "script": "483045022100df173cf465c319888425b5210e306dbf92e7a1bfe861626fb98306d9d6e676902204d5e8019f4e2a783e99a2cbe46fe8b33679bc1af583ceaa99418585bc99c2ad4014104e6e67eeb906914a27d075003367f3e10d11b693b9d30cdd912a70e94e3fdc59f50fd61c8e03ebac6f440fdcd4d1601fd699fb8f227cd1cf15e1cd07acc83c3a1"
    },
    {
      "weight": 896,
      "block_height": 654772,
      "relayed_by": "0.0.0.0",
      "out": [
        {
          "spent": false,
          "tx_index": 0,
          "type": 0,
          "addr": "16mxyzPp2teXpo87a2XASyzzMHFPRY1b9R",
          "value": 492400,
          "n": 0,
          "script": "76a9143f81bb4512b6b403caf3fe658c90a287359164b588ac"
        }
      ],
      "lock_time": 0,
      "size": 224,
      "block_index": 0,
      "time": 1604056755,
      "tx_index": 0,
      "vin_sz": 1,
      "hash": "720edc9c452709dbc1d49b93babaef8dd5fb813c3c9f63f880c1fff17061182f",
      "vout_sz": 1
    }
  ]
}

```

Now this is much more workable, isn't it? If we are only interested in inputs and outputs we can go a step further and use block explorers that simply provide decoded input and output information:

- [720edc9c452709dbc1d49b93babaef8dd5fb813c3c9f63f880c1fff17061182f](https://blockstream.info/tx/720edc9c452709dbc1d49b93babaef8dd5fb813c3c9f63f880c1fff17061182f)
- [720edc9c452709dbc1d49b93babaef8dd5fb813c3c9f63f880c1fff17061182f](https://www.blockchain.com/tx/720edc9c452709dbc1d49b93babaef8dd5fb813c3c9f63f880c1fff17061182f)

For now, let's put our focus on inputs/outputs only and use <https://blockstream.info> and <https://www.blockchain.com> explorers for that.

Inputs ⓘ

HEX

ASM

Index	0	Details	Output
Address	1AEqTxEgtCLsdG8bagtxKDW6tnULyV276V	Value	0.00600000 BTC
Pkscript	OP_DUP OP_HASH160 6556feefe150168a4708c2e4db5a5232b6a0910a OP_EQUALVERIFY OP_CHECKSIG		
Sigscript	3045022100df173cf465c319888425b5210e306bdbf92e7a1bfe861626fb98306d9d6e676902204d5e8019f4e2a783e99a2cbe46fe8b33679bc1af583ceaa99418585bc99c2ad401 04e6e67eeb906914a27d075003367f3e10d11b693b9d30cdd912a70e94e3fdc59f50f0d61c8e03ebac6f440fdc4d1601fd699fb8f227cd1cf15e1cd07acc83c3a1		
Witness	N/A		

Outputs ⓘ

Index	0	Details	Unspent
Address	16nnyzPp2teXpoB7a2XASyzzMNFPRY1b9R	Value	0.00492400 BTC
Pkscript	OP_DUP OP_HASH160 3f81bb4512b6b403caf3fe658c90a287359164b5 OP_EQUALVERIFY OP_CHECKSIG		

This transaction is as simple as imaginable, as it's just a P2PKH address transferring funds to another P2PKH. From it, we can conveniently get a look at how all P2PKH inputs and outputs will look like.

First, let's take a look at the input specifically:

Index	0	Details	Output
Address	1AEqTxEgtCLsdG8bagtxKDW6tnULyV276V	Value	0.00600000 BTC
Pkscript	OP_DUP OP_HASH160 6556feefe150168a4708c2e4db5a5232b6a0910a OP_EQUALVERIFY OP_CHECKSIG		
Sigscript	3045022100df173cf465c319888425b5210e306bdbf92e7a1bfe861626fb98306d9d6e676902204d5e8019f4e2a783e99a2cbe46fe8b33679bc1af583ceaa99418585bc99c2ad401 04e6e67eeb906914a27d075003367f3e10d11b693b9d30cdd912a70e94e3fdc59f50f0d61c8e03ebac6f440fdc4d1601fd699fb8f227cd1cf15e1cd07acc83c3a1		
Witness	N/A		

The Pkscript (also known as “locking script”) part here is taken from the output of an earlier transaction from which the Bitcoin originated, you will

be able to see it in the output of this transaction:

- [9ffe57579fb5268f0ca58cebcc3104be0775697e554644998e38fd267298f944](#)

The Pkscrip is visible in the blockchain.info raw transaction, there it is expressed as the following string:

76a9146556feefe150168a4708c2e4db5a5232b6a0910a88ac

This is still the exact same thing as the Pkscrip in the "Inputs" screenshot, only some code characters which stood for opcodes were converted to human-readable format:

HEX	ASM
OP_DUP	76
OP_HASH160	a9
OP_PUSHBYTES_20	14
	6556feefe150168a4708c2e4db 5a5232b6a0910a
OP_EQUALVERIFY	88
OP_CHECKSIG	ac

If you wish to find out the exact meaning of all these opcodes as well as check out what other opcodes exist and what codes they have in hexadecimal, you can see that here:

<https://docs.rs/syscoin/4.16.1/src/bitcoin/blockdata/opcodes.rs.html#85>

You may have noticed that the OP_PUSHCODE_20 opcode is omitted in the blockchain.info breakdown of the Pkscript, this is a rather standard practice as all this opcode does is prepare the next string of characters (which is usually the pubkey, pubkeyhash, signature...) to be pushed onto the stack. If this doesn't make much sense, that's okay, it's not that important anyways.

The Pkscript is a condition placed in the address that needs to be fulfilled to spend the output. In this case the condition is to prove that the you own the private key from which the public key hash (6556feefe150168a4708c2e4db5a5232b6a0910a) was derived from.

This is achieved by SigScript (also known as "unlocking script") which provides the public key itself (0250863ad64a87ae8a2fe83c1af1a8403cb53f53e486d8511dad8a04887e5b2352) and the signature (30440220230e80a311317e74fc11c15c5387af59825ee5e6f27833bd910f3a140c49db6202205e9720738db9f26d3dc1645bd759918e69438da5a87cef0cd599d4b6d326595f01), which was signed using the private key.

The Witness (additional data field added during the Segwit upgrade) remains empty for P2PKH inputs.

Outputs ⓘ

Index	0	Details	Spent
Address	36f9dsGq4sw1xy44ZAuMQ1AAiAyHZxBagF 	Value	0.00090000 BTC
Pkscript	OP_HASH160 367e3c2c31cb061606e5a812257fc153f2bef80e OP_EQUAL		

The output simply provides the next spending condition Pkscript that will be met in a future transaction.

Get Pagan's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

Subscribe

☐

Remember me for faster sign in

To simplify all of this, we can just say that P2PKH transactions will follow these patterns:

```
PkScript: OP_DUP OP_HASH160 <public key hash> OP_EQUALVERIFY
OP_CHECKSIG
SigScript: <signature> <public key>
Witness: N/A
```

Now that we know this pattern and are able to easily recognize P2PKH inputs and outputs, we can take a look at what patterns other address types leave.

P2SH is the trickiest to describe as technically, any conditions (within reason) can be seen in P2SH script, though typically it is used for multisignature transactions, the possibilities for P2SH addresses are near endless as they can use any combination of opcodes to create their locking

conditions and unlocking solutions. The P2SH pattern can be described as something like this:

```
PkScript: OP_HASH160 <redeem script hash> OP_EQUAL  
SigScript: OP_0 <condition solution> OP_PUSHDAT1 <redeem script>  
Witness: N/A
```

Here the “redeem script” is an arbitrary condition placed on the address. If this is a 2-of-3 multisig address the redeem script would look something like this:

```
Redeem script: OP_PUSHDAT2 <public key A> <public key B> <public key C>  
OP_PUSHDAT3 OP_CHECKMULTISIG
```

The “condition solution” is not a technical phrase, I just wished to show the pattern of P2SH transactions. If this was a multisig transaction, the “condition solution” would be at least 2 signatures created with the private keys from which the public keys from redeem script were derived from. So, assuming that the P2SH address is 2-of-3 multisig, the transaction parts would look like this:

```
PkScript: OP_HASH160 <redeem script hash> OP_EQUAL  
SigScript: OP_0 <signature A> <signature B> OP_PUSHDAT1 OP_PUSHDAT2  
<public key A> <public key B> <public key C> OP_PUSHDAT3  
OP_CHECKMULTISIG  
Witness: N/A
```

You can explore such a transaction here:

- 02b082113e35d5386285094c2829e7e2963fa0b5369fb7f4b79c4c90877dcd3d?
output:0
- eeab3ef6cbea5f812b1bb8b8270a163b781eb7cde10ae5a7d8a3f452a57dca93
?input:0

The remaining address types follow really similar patterns as these 2 mostly with only some parts moved around. Here are their patterns and some transactions to explore:

P2WPKH:

PkScript: *OP_0 <public key hash>*
SigScript: *N/A*
Witness: *<signature> <public key>*

Example transaction:

452d61cd94212cb59db58169a69ba223745b099d58e8fc51ddedee16e1efdc14

P2WSH:

PkScript: *OP_0 <redeem script hash>*
SigScript: *N/A*
Witness: *<condition solution> <redeem script>*

Example transaction:

cae0cfd93f8defec67795903c1d6bbbc61f9eac6892edb9eb1d104fb7ee47a877

P2WPKH nested in P2SH:

```
PkScript: OP_HASH160 <redeem script hash> OP_EQUAL  
SigScript: OP_0 <public key hash>  
Witness: <signature> <public key>
```

Example transaction:

43b31221a55de822b4503e667c29a98fbab5fcf6b26e2291658ff24e6741ea29

P2WSH nested in P2SH:

```
PkScript: OP_HASH160 <P2SH redeem script hash> OP_EQUAL  
SigScript: OP_0 < P2WSH redeem script hash>  
Witness: <condition solution> <P2WSH redeem script>
```

Example transaction:

36ff5db60887547464cf7fab50df6539332751bccec03cb6d55fb904d128f879?
expand

The reason I just went over existing address types is to show just how much variety there is between them, especially if you include all the variety that P2SH, P2WSH and P2WSH nested in P2SH can have, and just how easy it is to recognize and analyze these addresses. This variety is in itself a privacy risk, the more address types exist the easier it is to identify individual wallets and consequently individual users. Ideally, Bitcoin should have only one address, but the cat is out of the bag now. In any case, if you are interested in learning more about Bitcoin scripts and addresses, I'd recommend checking out these links:

<https://blog.susanka.eu/types-of-bitcoin-transactions-part-i/>

<https://blog.susanka.eu/types-of-bitcoin-transactions-part-ii-segwit/>

<https://en.bitcoin.it/wiki/Script>

There are a few more things that can distinguish a transaction I'd like to briefly cover in this section before moving on:

RBF (Replace-by-Fee):

RBF is a feature of Bitcoin transactions that indicates that the transaction is replaceable while it is still in the mempool. A few wallets offer such a feature while most do not, making the few that support the feature stick out.

A transaction can signal that it supports RBF in it if any of its inputs have a Sequence number that's lower than 0xffffffff — 1. Here is an example of such a transaction:

- [17e364a79eca847d077c0311eed61323f38cb39cb0e9d3450979c120bfea40bb](#)
- [1e175480a3945a0c567ddf56b03c53e258179ac7912e08d0e4dfd9b2907ac684](#)

```

{
  "ver":2,
  "inputs":[
    {
      "sequence":0,
      "witness":[],
      "prev_out":{
        "spent":true,
        "spending_outpoints":[
          {
            "tx_index":0,
            "n":0
          }
        ],
        "tx_index":0,
        "type":0,
        "value":274870855,
        "n":1,
        "script":"00204886ed4694c4d6c3a145c6c14ab796a6b01f840d20449a32305ff2efdb6fb166"
      },
      "script":""
    }
  ],
  "weight":902,
  "block_height":660606,
  "relayed_by":"0.0.0.0",
  "out":[
    {
      "spent":false,
      "tx_index":0,
      "type":0,
      "addr":"1LByTxSS8zTe5Z58inXdZAga9YTNUTu2Yr",
      "value":1040000,
      "n":0,
      "script":"76a914d27daf14ff30b94d0e8b2ffd2fd7aeddea784a8288ac"
    },
    {
      "spent":false,
      "tx_index":0,
      "type":0,
      "addr":"18oK6iXPv4dWnPjCX5dtS51rxA8YaNvMy1",
      "value":390000,
      "n":1,
      "script":"76a914558b1f6532e25a2ddff2354619212a77bf03fa0888ac"
    },
    {
      "spent":false,
      "tx_index":0,
      "type":0,
      "value":273378605,
      "n":2,
      "script":"00204886ed4694c4d6c3a145c6c14ab796a6b01f840d20449a32305ff2efdb6fb166"
    }
  ],
  "lock_time":0,
  "size":416,
  "rbf":true,
  "block_index":0,
  "time":1607507033,
  "tx_index":0,
  "vin_sz":1,
  "hash":"1e175480a3945a0c567ddf56b03c53e258179ac7912e08d0e4dfd9b2907ac684",
  "vout_sz":3
}

```

You can read more about opt-in RBF [here](#):

Locktime:

Locktime sets the earliest blockheight or time that a transaction can be mined. Most wallets and exchanges don't use this feature, thus the few that do are easier to identify. The locktime can be seen in the raw transaction data, if the locktime number is above or equal to 500000000 it is interpreted as Unix time (seconds from 1st January 1970), if it's below this number it is interpreted as block height. Here are a few example transactions with locktime enabled:

- 0fa5a64bd1ac65233c3883b010bf641988fa3f4d4755212c43243837920ce051
- 5343bd9702c0f5b0a37c86c32e3cc375c24fd61b4d392b4ce48e2d4a99593cb4

```

    "script": "0fa5a64bd1ac65233c3883b010bf641988fa3f4d4755212c43243837920ce051",
  },
  {
    "lock_time": 660618,
    "size": 1790,
    "block_index": 0,
    "time": 1607514647,
    "tx_index": 0,
    "vin_sz": 1,
    "hash": "5343bd9702c0f5b0a37c86c32e3cc375c24fd61b4d392b4ce48e2d4a99593cb4",
    "vout_sz": 49
  }

```

Methods to write information in a transaction:

The very first Bitcoin block came with the famous and ever-true message:

General info

0

1

Hash
00000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

Mined on 2009-01-03 18:15 (12 years ago) Miner Unknown

Coinbase data EThe Times 03/Jan/2009 Chancellor on brink of second bailout for

		USD	BTC
Transaction count	1	Fee per kB	0.00000000 BTC
Witness tx count	0	Fee per kWU	0.00000000 BTC
Input count	1	Output count	1
Input total	0.00000000 BTC	Output total	50.00000000 BTC
Fee total	0.00000000 BTC	Coindays destroyed	0.00
Generation	50.00000000 BTC	Reward	50.00000000 BTC

Technical details

Difficulty 1

Size 285

Weight (weight units) 1,140

Stripped size 285

Version 1₁₀ 1₁₆

Version [bits] 00000000000000000000000000000001₂

Median time 2009-01-03 18:15:05

Merkle root 4a 3b

For developers [API docs](#)

Alternative explorers [BTC](#)

[Click to see more ↓](#)

Transactions included in this block

Block # ▼	Hash	Inputs #	Outputs #	Coindays destroyed	Output (BTC)	Output (USD)	Transaction fee (BTC)	Transaction fee (USD)	Fee/kB (BTC)	Size (kB)
0	4a 3b	1	1	0.00	50.00000000	0.50	0.00000000	0.00	0.00000000	0.204

<https://blockchair.com/search?q=4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b>

Since then, lots of messages were written in the Coinbase data, but that is not the only way messages can be left in the blockchain, they can also be left instead of outputs if OP_RETURN opcode is used:

OP_RETURN

Decoded

j!Xuemei, will u marry me? -Shangqi

Senders

⏪ ⌚ 0.01926180 BTC

Recipients

36dWhFGKY15KbZ5NsJFF37CBYu8WJJk63V

0.01314000 BTC Unspent

3AtmyLjWqb1xxAokXxQGfZNo5VWobaEGE

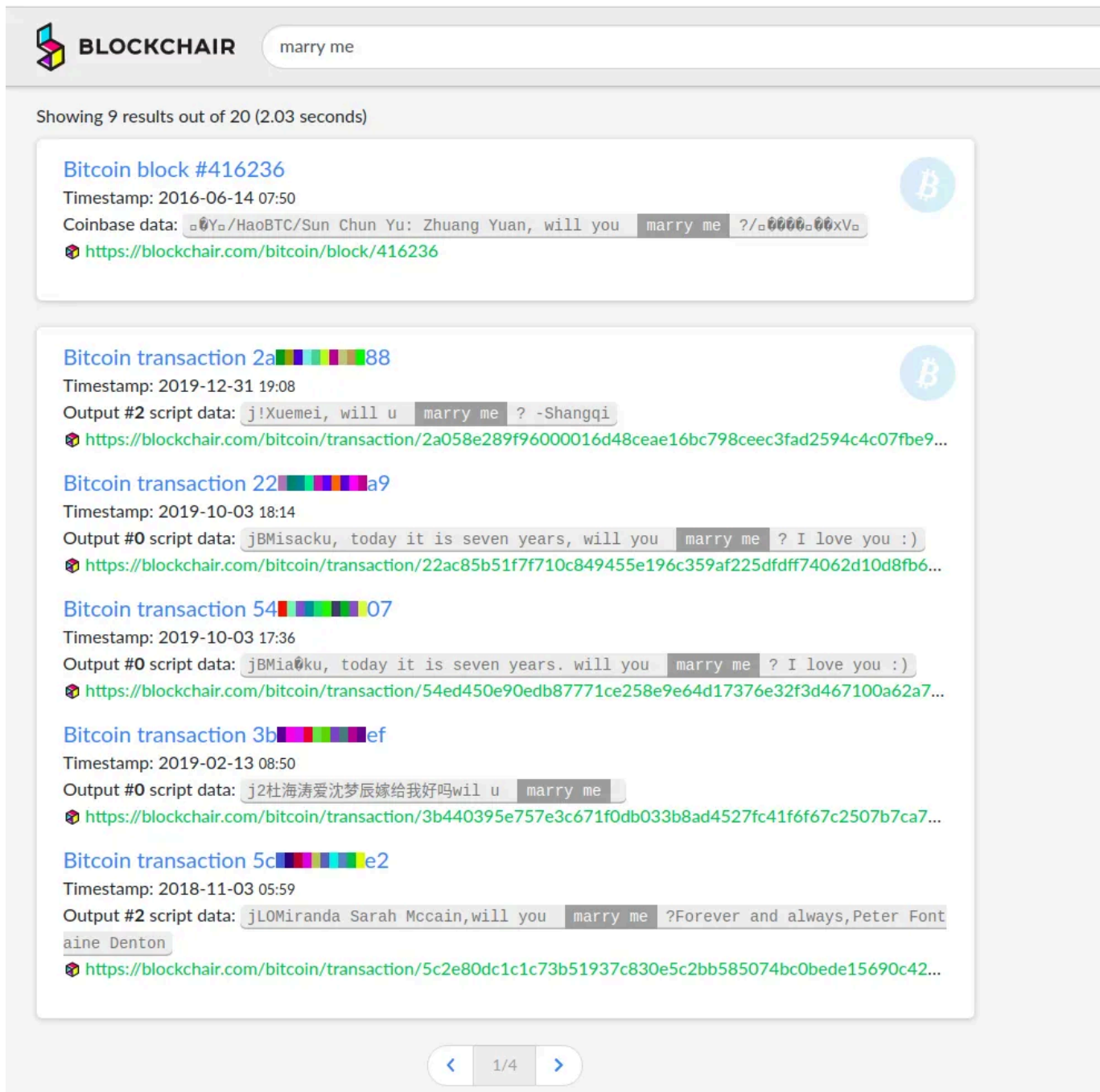
0.00610364 BTC Unspent

OP_RETURN d-9c751460b640e3a6ac1425287c14218

0.00000000 BTC Unspent

<https://blockchair.com/bitcoin/transaction/2a058e289f96000016d48ceae16bc798ceec3fad2594c4c07fbe935e38e32688>

You can actually see a bunch of such messages if you just search for words in <https://blockchair.com/>:



BLOCKCHAIR marry me

Showing 9 results out of 20 (2.03 seconds)

Bitcoin block #416236

Timestamp: 2016-06-14 07:50

Coinbase data: a0Y0/HaoBTC/Sun Chun Yu: Zhuang Yuan, will you marry me ?/a0000a00xV0

<https://blockchair.com/bitcoin/block/416236>

Bitcoin transaction 2a88

Timestamp: 2019-12-31 19:08

Output #2 script data: j!Xuemei, will u marry me ? -Shangqi

<https://blockchair.com/bitcoin/transaction/2a058e289f96000016d48ceae16bc798ceec3fad2594c4c07fbe9...>

Bitcoin transaction 22a9

Timestamp: 2019-10-03 18:14

Output #0 script data: jBMisacku, today it is seven years, will you marry me ? I love you :)

<https://blockchair.com/bitcoin/transaction/22ac85b51f7f710c849455e196c359af225dfdf74062d10d8fb6...>

Bitcoin transaction 5407

Timestamp: 2019-10-03 17:36

Output #0 script data: jBMia0ku, today it is seven years. will you marry me ? I love you :)

<https://blockchair.com/bitcoin/transaction/54ed450e90edb87771ce258e9e64d17376e32f3d467100a62a7...>

Bitcoin transaction 3bef

Timestamp: 2019-02-13 08:50

Output #0 script data: j2杜海清爱沈梦辰嫁给我好吗wil u marry me

<https://blockchair.com/bitcoin/transaction/3b440395e757e3c671f0db033b8ad4527fc41f6f67c2507b7ca7...>

Bitcoin transaction 5ce2

Timestamp: 2018-11-03 05:59

Output #2 script data: jLOMiranda Sarah McCain,will you marry me ?Forever and always,Peter Fontaine Denton

<https://blockchair.com/bitcoin/transaction/5c2e80dc1c1c73b51937c830e5c2bb585074bc0bede15690c42...>

< 1/4 >

If you want to learn more about how these sort of transactions can be created, check out [this](#) article.

Another little way to leave your imprint on the blockchain is through the use of [vanity addresses](#).

CoinJoins and PayJoins

An in-depth explanation of Coinjoins and Payjoins would really need an article of its own (which I will probably do someday), so I will have to simplify this section a lot and leave out a lot of important details. For the time being, if you wish to learn more about Coinjoins and Payjoins I'd recommend checking out these articles:

<https://en.bitcoin.it/wiki/CoinJoin>

<https://en.bitcoin.it/wiki/PayJoin>

<https://joinmarket.me/blog/blog/payjoin/>

In short, Coinjoin is simply a transaction designed to break the common-input-ownership heuristic by having the inputs and outputs of the transaction belong to multiple entities. If a blockchain analyst is unable to deduce which inputs and outputs belong to which entities their investigation is effectively moot. Coinjoin transactions usually look something like this:

General info

Hash (txid) `b3dcc5d68e7ba4946e8e7fec0207906fba89ccb4768112a25d6e6941f2e99d97`

Block Id `564929` 90444 confirmations

Time (UTC) `2019-02-27 21:12` (2 years ago)

[PDF receipt](#) USD ☒ BTC

Input total	39.43889378 BTC	Output total	39.43872236 BTC
Transaction fee	0.00017142 BTC	Fee per byte	1 satoshi
Replace-by-fee (RBF) enabled?	NO	Fee per vbyte	2 satoshi

Technical details

Input count / Output count 59 / 120

Size 12,464

Coindays destroyed 26.45

For developers [API docs](#) [Raw tx](#)

Alternative explorers [BRC](#) [G](#)

[Click to see more](#)

Senders

- [0.1989564 BTC](#) `bc1q9yr5wv7h4ehz2kpjncuyk7a8qgwhuzuyqep66u`
- [0.09950717 BTC](#) `bc1qa3g3sq74fgevuytmr4rh6sfnf8tchw29j7km2h`
- [0.09980140 BTC](#) `bc1qrkactec8g7eltx24veges5twijq2rphl4dquxj`
- [0.00946702 BTC](#) `bc1qrjse7manzmsd0qhphte28qmdxcerahywv46l`
- [0.09942481 BTC](#) `bc1quj6yl7umf5qta6vrye836mgxp4radgfnkc3rw`
- [0.00365469 BTC](#) `bc1q5cqcqd7nhw9lh7wgjkuwoe2wknsfzg86nwnh`
- [0.09873783 BTC](#) `bc1qx5jqvx2p5ehuskn6tythlnn8mpwqOedv07myt8`
- [0.09950717 BTC](#) `bc1qxflv308ed63ez4a0ss4pcjtw6uv23k4xzlusk`
- [0.89526973 BTC](#) `bc1qk0z8g20gz7w0zy2hnqj50sgl85ml7uud7lmvwa`
- [0.39799256 BTC](#) `bc1qnece3jcw70ly0gmsnqfgc5ctzvc7kfy6rjthp8`
- [0.09953753 BTC](#) `bc1qjg26fd8szdfmxz2eaw36ckhlhg0ajfkmydquq`
- [0.10036188 BTC](#) `bc1qsq8gxy3myucy6h7ea4uerqclgpyqmywn75anm`
- [0.09960684 BTC](#) `bc1qtjmmvu8cy8wav4rqh8qlc3tzwgyp83h50knkat`
- [0.39798904 BTC](#) `bc1q85rvjk0psaq9vr4qf8j5e4634flp5n7cstcw7r`
- [0.09950717 BTC](#) `bc1qczn06rrur96fwcpx23t5sn5wluhmk50mnsccd`
- [0.39799256 BTC](#) `bc1q550l9nmgmjmv5r23gnjtxcpsckhq407slfv98t`
- [0.09950726 BTC](#) `bc1qah20mjjls72d0l7xpaphgztm6zhd03dnk63cer`
- [0.39798988 BTC](#) `bc1q5u9nhqxd6yh0wjrqswvceut62qmwjwz05s7`
- [3.99377419 BTC](#) `bc1qxaqg95v3lvqas9h52a05mr3glnc4vq352x707`
- [0.39798904 BTC](#) `bc1qt0m346ls8c0n65whdnaedqgfkzxtt3se239`
- [0.30326294 BTC](#) `bc1q4smsm5mfjrvvqt0ax8pfcxc3k60n3xk9p4gg0`
- [0.79597808 BTC](#) `bc1qea4sncwsk5v96ex6384pnwvwtmalfll7hdplox`
- [0.26128111 BTC](#) `bc1qmgztvnyvrpe8ujcuq3pj4r38u63rg7zrljug7`
- [0.09950717 BTC](#) `bc1qad47x89lgwnu9ca43fj9cq27eme8phqs0u296`

Recipients

- [0.09950959 BTC](#) `bc1q0fazqzfln062nd4prkjrrpf0ws53xmttd30wgf`
- [0.09950959 BTC](#) `bc1qdn92wlmgnnee6c0zku86qvsqvlpsmxz03y9k8a`
- [0.09950959 BTC](#) `bc1qgvqq8e4v0jh6jwa70k6qqfeft065ehexm75d`
- [0.09950959 BTC](#) `bc1qwcsxytvr9p5799laqd7wf20y3u20htxld6zysrh`
- [0.09950959 BTC](#) `bc1qwnx73cwwfl6efag6m46kn55bhz4qmu3hefm`
- [0.09950959 BTC](#) `bc1q0p2mnh7pjjeqrddt6u36lrr5zh6f5zwynmmn`
- [0.09950959 BTC](#) `bc1qcu4x30tevtq0vqgy3lul3nsh4wvqevm0773r`
- [0.09950959 BTC](#) `bc1q4vglpjij4az4dz6tsm7jgl84y58y0hk43g2a9u`
- [0.09950959 BTC](#) `bc1qqm3up8wu9uf7xhdv07dwzweej9sj9se0w9hrhg8`
- [0.09950959 BTC](#) `bc1qn757jwr7vn7fxcyse6k0dwqp4s5zfjh0twnfnpk`
- [0.09950959 BTC](#) `bc1q5s6u6ux3glk6zyamyhx3ner875zf9wtglqkjp8`
- [0.09950959 BTC](#) `bc1qglmx22avxywnqwyOqg7x72eytccm650r0ey45f`
- [0.09950959 BTC](#) `bc1qj2n05622r6adkux6xm973zx6vkm62727yct7zy`
- [0.09950959 BTC](#) `bc1qwk93ylez3wnf2zpu4zvtwkeg8l609yk8mc3xtz`
- [0.09950959 BTC](#) `bc1q0hpke5zdc908b3exd3qa3ez5gyq8c63zysn8d4`
- [0.09950959 BTC](#) `bc1qerl5hulcman8y7gsz5u8zd40lacdngvqzm7un`
- [0.09950959 BTC](#) `bc1qz4zdwjx9t3w0pk8plme34pws3tx69jf9wnc9x6`
- [0.09950959 BTC](#) `bc1qfp42lnrs69zbxwhcvt6hm48lcxahetd80jdc`
- [0.09950959 BTC](#) `bc1q0nc6kthruyh6zm44283h7h0nd6edqac33yp38`
- [0.09950959 BTC](#) `bc1q9suevveu477rh6pfg252rwdwcyf6mpjd58z8q`
- [0.09950959 BTC](#) `bc1qgusxe8735spx3djmsdqxg2rlvaqkcfh3jshzjz`
- [0.09950959 BTC](#) `bc1q54jh0eq9lq6xjrs2mk5k23crtm8yrhxj8n45w3`
- [0.09950959 BTC](#) `bc1q7h7u4w0464lf4v8xdgz6r2gelujx9v2vz27e75`
- [0.09950959 BTC](#) `bc1q9gz27yc2t7nwwcyvlyfrpz77agktjw9rgdadda`

Coinjoin transactions will usually have a large number of inputs and outputs belonging to many different entities and the value of the outputs will be nearly identical (the value is identical to avoid amount-correlation-heuristic). However, this is not the only way they can look, the following is also a CoinJoin:

The screenshot displays a Bitcoin transaction page on blockchair.com. The main section shows transaction details for hash 323df21f0b0756f98336437aa3d2fb87e02b59f1946b714a7b09df04d429dec2, which has 86540 confirmations and was created on 2019-05-05 09:03 (2 years ago). A PDF receipt is available. The transaction summary shows an input total of 0.25030000 BTC, an output total of 0.25000000 BTC, a transaction fee of 0.00030000 BTC, and a fee per byte of 33 satoshi. The Replace-by-fee (RBF) is disabled. A technical details sidebar on the right shows 5 inputs and 5 outputs, a size of 911, and 0.49 coindays destroyed. Below the main details, a 'Senders' and 'Recipients' section lists the addresses and amounts involved in the transaction.

Senders		Recipients	
0.05010000 BTC	bc1qgww9hn979rex0rhfp3cknfdkyrgk04493gclvv	0.05000000 BTC	bc1qqmmc3s46efrdq0jghf8l8jg0xw37exgne6q3k
0.05000000 BTC	bc1qtgstl44zcltvd62s7kctxtqjht0zmufhqr9f	0.05000000 BTC	bc1qrjh3lq5rmvw4umuslgpqcnyhl9crkv45z03gg
0.05010000 BTC	bc1q528pyda90y4j9nuqrcm65rlxz9tdjtn208dqm	0.05000000 BTC	bc1qywwz0hhkvhj3x6ef69x0rewz6vjaruxqva0fz
0.05000000 BTC	bc1qnuu6u8d4rm7kavmrnnpn36lts5gajutf34rtfv	0.05000000 BTC	bc1qfc06trnvvg304xvkfzsdxdrcy3z6ef4flsxt
0.05010000 BTC	bc1q88yvt3sqx7hucswejmna7pvp4ga7j0ukfppwq	0.05000000 BTC	bc1qhevgmqzrlxqahama5f4dydrem3uf4vaj43kg2
		Unspent	

<https://blockchair.com/bitcoin/transaction/323df21f0b0756f98336437aa3d2fb87e02b59f1946b714a7b09df04d429dec2>

At the moment, there are 3 main wallets that offer the CoinJoin feature:

Samourai wallet: <https://samouraiwallet.com/>

Wasabi wallet: <https://wasabiwallet.io/>

Joinmarket: <https://github.com/JoinMarket-Org/joinmarket-clientserver>

The Coinjoins created by each of these wallets have some unique technical characteristics and thus it may be possible to recognize which CoinJoin belongs to which wallet. In general, recognizing CoinJoin transactions isn't much of a problem, however, solving CoinJoin transactions themselves is a different deal entirely. At the moment, finding ties between inputs and outputs (whether in-between several inputs/outputs or inputs with outputs) is only possible if the CoinJoin was constructed incorrectly (such as addresses being reused for inputs and outputs) or if the entities involved have made privacy mistakes after the CoinJoin transaction (for example, by using their new "clean" CoinJoin outputs in a transaction with their old "unclean" traceable outputs). This is admitted even by Europol in the

following 2020 report:

<https://www.tbstat.com/wp/uploads/2020/06/Europol-Wasabi-Wallet-Report.pdf>

The absolute best free tool for analyzing if CoinJoins were constructed correctly and if the entities involved made any mistakes, later on, is the <https://www.kycp.org/#/> blockchain explorer. I will save myself sanity by not explaining how it works as thankfully there is already a very comprehensive guide on how to use it here:

<https://medium.com/samourai-wallet/knowning-your-coin-privacy-using-kycp-org-7b3b4385d8b>

To summarize, the CoinJoin transaction remains the most powerful privacy tool available to Bitcoin users with its only major drawback being that these type of transactions are easy to identify*

This flaw of CoinJoin's is addressed partially by the PayJoin transaction. Payjoin transactions are, in essence, payment CoinJoin transactions that are indistinguishable from regular 2 output transactions here is an example of a Payjoin transaction:

General info

Hash (txid) 14ff5e5c9206f7722b0ed421d26f1cb003c523070c4882150073817683bd2dda

Block Id 642767 18461 confirmations

Time (UTC) 2020-08-08 11:11 (4 months ago)

PDF receipt

USD ☐ BTC ☒

Input total	0.00504941 BTC	Output total	0.00502684 BTC
Transaction fee	0.00002257 BTC	Fee per byte	6 satoshi
Replace-by-fee (RBF) enabled?	NO	Fee per vbyte	11 satoshi

Blockchair Chrome Extension

Blockchair search now in your browser

Get [→](#)

Technical details

Input count / Output count 2 / 2

Size 372

Coindays destroyed 0.02

For developers [API docs](#) [Raw tx](#)

Alternative explorers [btc](#) [bch](#)

[Click to see more](#)

Senders

[← ⓘ](#) 0.00279088 BTC [bc1q88v04nxtaj658lzkcywq3c3lu0nprdd3p9rq97](#)

[← ⓘ](#) 0.00225853 BTC [bc1qx9g8zca8qw76sdsvapctrn45qfk4dvwzpmdd5z](#)

Recipients

[bc1qnh4k7i7hp8uqvird352ttkug630nqzxt23ekt](#) 0.00023596 BTC [ⓘ →](#)

[bc1qjgtzyvpj74q6kaw6s8lrled6he5jc6r8xlnxy](#) 0.00479088 BTC [ⓘ →](#)

<https://blockchair.com/bitcoin/transaction/14ff5e5c9206f7722b0ed421d26f1cb003c523070c4882150073817683bd2dda>

Looks like just a regular transaction, right? It would be reasonable to assume that both inputs belong to the same owner and the second output (the 0.004... BTC one) is the destination address, while the first output is the change. However, in reality, it is a payment in which one party paid the other exactly 0.002 BTC. Input one and output 2 is owned by a merchant who is getting paid 0.002 BTC in this transaction (we can see that output 2 amount equals input 1 amount plus 0.002 BTC), while input 2 and output 1 belong to the customer that paid 0.002 BTC to the merchant (because output 1 equals input 2 minus 0.002 BTC and the transaction fee). These are 2 wildly different interpretations of the same transaction that would surely cause a headache for any Bitcoin analyst.

This transaction was actually posted as a fun “guess the amount game” by its owner on Twitter:

https://twitter.com/BitcoinQ_A/status/1335517903127539713

The drawback of the PayJoin transaction scheme is that it only works with payments and thus its use case is limited. Nevertheless, since PayJoins are indistinguishable from normal transactions so their adoption is important for the Bitcoin network as a whole as they can plant seeds of uncertainty in blockchain forensic investigations.

* It's a lot more complex than that, but again that's beyond the scope of this article.

Other misc identification patterns

In this section, I would like to briefly mention some of the other various identification patterns that are too obscure or technical to deserve a section of their own, but may nonetheless be interesting to know about:

BIP 69 Lexicographical index of transaction inputs and outputs — there is a proposed standard for ordering the inputs and outputs of transactions for BTC wallet clients. While this standard seeks to improve the privacy of transactions, some wallet implements it while others do not, meaning that it can be another clue for wallet fingerprinting:

https://en.bitcoin.it/wiki/BIP_0069

<https://transactionfee.info/charts/transactions-bip69-compliant/>

Low-R value signature — BTC core wallet saves transaction space by shortening the signature just a little bit. No other wallets do this (at least according to <https://en.bitcoin.it/wiki/Privacy>):

<https://github.com/bitcoin/bitcoin/pull/13666>

Uncompressed/compressed public keys — when Satoshi was creating Bitcoin, they did not realize that BTC public keys can be compressed to save

space without any side effects. Eventually, people realized this and wallets started using compressed public keys, but still, there are some that use uncompressed ones, so that sets them apart:

<https://bitcoin.stackexchange.com/questions/3059/what-is-a-compressed-bitcoin-key>.

<https://www.ntirawen.com/2019/03/bitcoin-compressed-and-uncompressed.html>

Coin selection — being able to select which specific UTXO's you wish to spend. Several wallets offer this functionality (such as Bitcoin Core wallet):
<https://nopara73.medium.com/coin-control-is-must-learn-if-you-care-about-your-privacy-in-bitcoin-33b9a5f224a2>

Sybil attack — an old type of attack on peer-to-peer systems where the attacker pretends to be several different entities to disrupt the privacy or security of the system. In Bitcoin, this kind of attack is most commonly associated with network forensics, but since CoinJoins are peer-to-peer systems they can also be threatened by Sybil attacks. Oxt.research team has observed this kind of attack be inadvertently executed on a CoinJoin wallet and were able to track the entity they were targeting because of it. If you are interested in learning more I'd recommend checking out their research here:
<https://research.oxt.me/publications>

Sidechains — a mechanism that allows moving Bitcoins to another, independent blockchain. The other blockchain may have very different privacy features. In practice at this time, the only functioning sidechain for the Bitcoin is the Liquid sidechain. You may read more about sidechains in general here:

<https://en.bitcoin.it/wiki/Sidechain>

Handmade CoinJoin — CoinJoin transactions can be hand-made without a special wallet just using Raw Transactions API. This provides a degree of flexibility:

https://en.bitcoin.it/wiki/Raw_Transactions

Taint analysis — a simplistic blockchain analytic technique where the analyst assumes that every address that touches the address they are tailing (the first “tainted” address) via transaction is also “tainted”. This sort of assumption may be useful when investigating darknet markets but does not take into account transfer of ownership and thus is really limited in its effectiveness.

Read more here:

<https://bitcoin-class.weebly.com/taint-analysis.html>

Centralized mixers — before CoinJoins came along it was popular to mix bitcoins in a custodial manner. There is a lot of variety on how centralized mixer privacy works (and in many cases don't work), however, generally, the privacy from a centralized mixer will not be better than the one provided by decentralized CoinJoin. Also, you must trust the mixer with your money while you are using it. Personally, I would not do that no matter how hard the centralized mixer pinky promises they will not spy on you or give your information to the FBI/CIA/KGB/NSA/M16... Mixers are usually quite shady and could only be found in the darknet. You can read more about them here:

https://en.bitcoin.it/wiki/Mixing_service

<https://medium.com/cobo-vault/bitcoin-mixing-a-brief-research-into-centralized-vs-decentralized-390ea83fa6ad>

The future of privacy:

Bitcoin privacy and is constantly changing with the introduction of new tools that serve to make the bitcoin analytics harder/easier at the same time the

Bitcoin network is also gradually changing as a whole. Although the future can never be fully predicted, we can do our best to make a few general guesses as to how space will evolve and what technologies will influence this change:

Lightning Network — as more and more people (hopefully) start using the lightning network its impact on the privacy of Bitcoin will be more and more noticeable. Only the initial and exit transactions of the lightning network are visible on the Bitcoin ledger while the possibly thousands of intermittent transactions between initial and exit one all happen off-chain. Sadly, I do not know enough about the lightning network to be able to determine its full impact on Bitcoin analytics, but it will for sure have one. But I will leave a few links to articles from which you may be able to begin forming your own opinion:

https://en.bitcoinwiki.org/wiki/Lightning_Network

<https://medium.com/suredbits/lightning-network-101-privacy-245d98210ab>

<https://arxiv.org/abs/2003.12470>

<https://www.coindesk.com/bitcoin-lightning-privacy-point-timelock-contracts-ptlcs>

Coinswap — there is a new privacy tool being developed by one of the people behind JoinMarket and the person who wrote the massive Bitcoin wiki's privacy page, the lord and savior of Bitcoin privacy — Chris Belcher. Coinswap promises to provide the holy grail of Bitcoin privacy, coinjoin which is both unidentifiable and convenient to use for everyone, not just merchants and buyers. More information about the CoinSwap protocol can be read here:

<https://gist.github.com/chris-belcher/9144bd57a91c194e332fb5ca371d0964>

Taproot and on-chain changes — the Bitcoin protocol changes from time to time, the last major change was the Segwit softfork. The next will be the Taproot softfork. Among other goals, this softfork seeks to improve the privacy of Bitcoin script. Once again, I am ashamed to admit I do not know enough about this softfork so, for the time being, I will defer to the Bitcoin wiki and a few great articles where you can read more about Taproot:

https://en.bitcoin.it/wiki/BIP_0341

<https://medium.com/interdax/how-will-schnorr-signatures-benefit-bitcoin-b4482cf85d40>

<https://medium.com/interdax/what-is-taproot-and-how-will-it-benefit-bitcoin-5c8944eed8da>

<https://medium.com/interdax/what-is-tapscript-and-how-will-it-benefit-bitcoin-96fbb43a7169>

This concludes the second part of the “Free Blockchain Forensics” article, with this information anyone can certainly conduct blockchain forensics investigations on their own and be quite successful at them. In the third part of the article, I conducted a mini-investigation of my own to demonstrate how that could work. Check it out if you’re interested ;)

Also, Read

- The Best Crypto Trading Bot
- Uniswap API — How to get Uniswap data?
- AAX Exchange Review | Referral Code, Trading Fee, Pros and Cons
- Deribit Review | Options, Fees, APIs and Testnet
- FTX Crypto Exchange Review
- Bybit Exchange Review
- The Best Bitcoin Hardware wallet

- [Crypto Copy Trading Platforms](#)
- [Bitsgap vs 3Commas vs Quadency](#)
- The Best [Crypto Tax Software](#)
- [Best Crypto Trading Platforms](#)
- Best [Crypto Lending Platforms](#)
- [Ledger Nano S vs Trezor one vs Trezor T vs Ledger Nano X](#)
- [BlockFi vs Celsius vs Hodlnaut](#)
- [Bitsgap review](#) — A Crypto Trading Bot That Makes Easy Money
- [Quadency Review](#)- A Crypto Trading Bot Made For Professionals
- [PrimeXBT Review](#) | Leverage Trading, Fee and Covesting
- [Altrady review](#)
- [Ellipal Titan Review](#)
- [SecuX Stone Review](#)
- [BlockFi Review](#) | Earn up to 8.6% interests on your Crypto
- [Best Crypto APIs](#) for Developers
- [Best Blockchain Analysis Tools](#)
- [Crypto arbitrage](#) guide: How to make money as a beginner
- Top [Bitcoin Node](#) Providers
- Best [Crypto Charting Tool](#)
- What are the [best books to learn about Bitcoin?](#)

Get Best Software Deals Directly In Your Inbox

Best Software Deals

Exclusive software deals directly in your inbox

I'AM GAME

Bitcoin

Forensics

Cryptocurrency

Blockchain

Crypto



Published in Coinmonks

144K followers · Last published 8 hours ago

[Follow](#)

Coinmonks is a non-profit Crypto Educational Publication. Other Project — <https://coincodecap.com/> & Email — gaurav@coincodecap.com



Written by Pagan

Follow

30 followers · 37 following

Just a hobbyist crypto enthusiast :)

